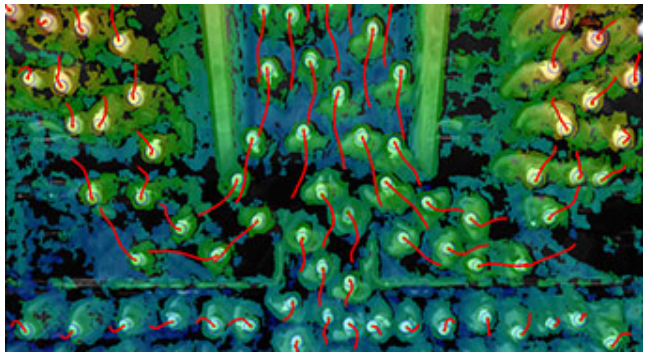
JÜLICH
FORSCHUNGSZENTRUM

Fire Simulation

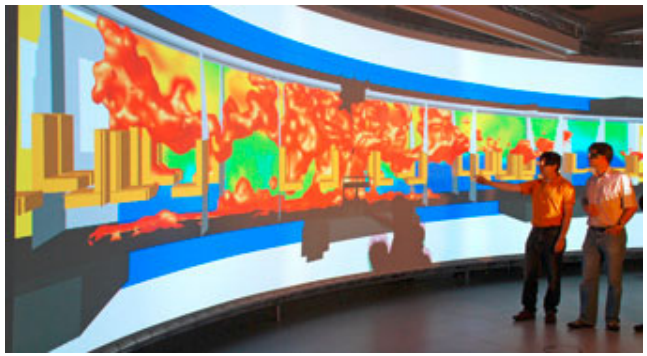
June 15, 2016 | Marc Fehling | Jülich Supercomputing Centre (JSC)

JSC - Division Civil Security and Traffic (CST)

- **Pedestrian dynamics**
  - Experiments, modeling, simulation
  - Civil Security at major events
  - Assessment of traffic installations
- **Traffic**
  - Large–scale traffic simulation



- **Fire simulation**
  - Parallel and adaptive methods for fire simulations
  - Safety in underground stations
  - PIV experiments for validation



Fire Group Activities: Simulation

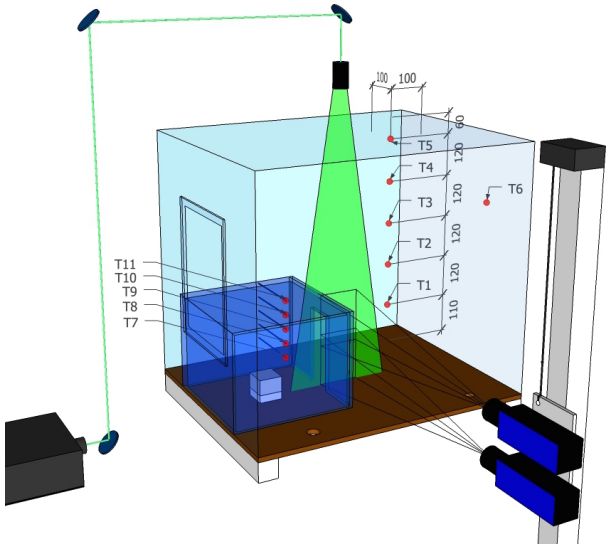
- Simulations of smoke and fire spread
  - Modeling of combustion, radiation, heat transfer, and smoke propagation
  - Use of common software (FDS, FireFOAM), contribute to its development
  - Write own software with various specializations
- For use on desktop machines, GPU nodes, and supercomputers

Jureca Cluster

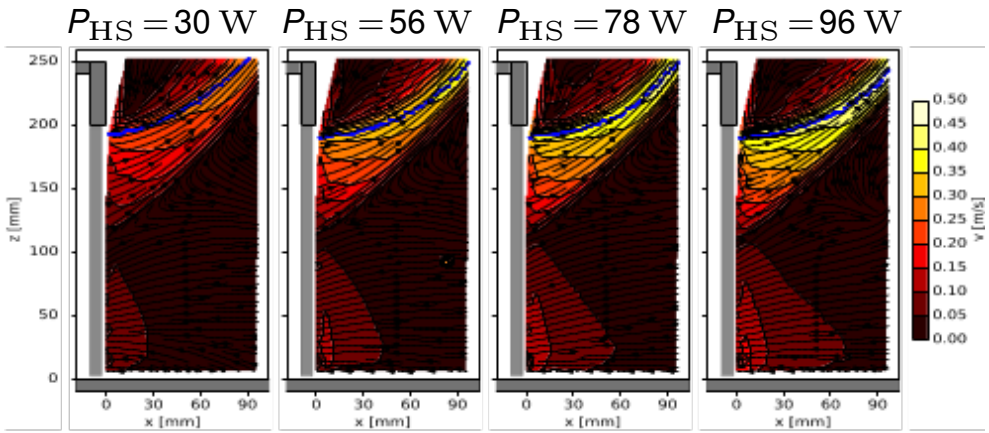
Nvidia Tesla K80

Fire Group Activities: Experiments

- Particle Image Velocimetry (PIV) experiments
  - Electrically heated source as driver
  - Seeding particles for flow visualization
  - Laser for particle illumination, cameras for recording
  - Temperature recording with thermocouples
- Using both commercial and self–written software for evaluation and simulation/prediction



Door width 40mm

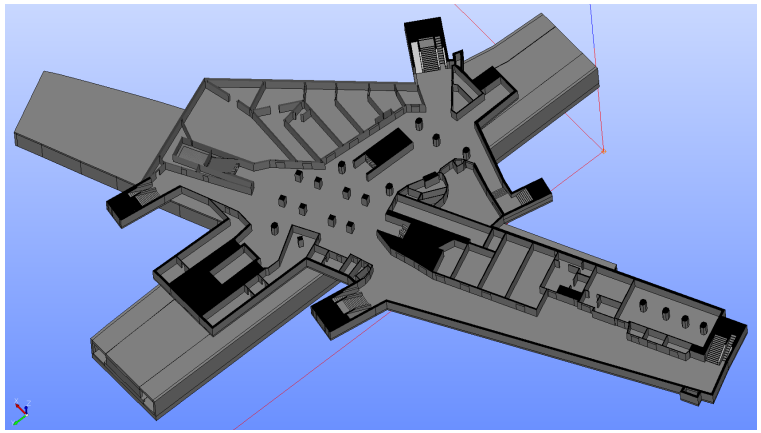


Fire Group Activities: Project Orpheus

ORPHEUS

Optimierung der **R**auchableitung und **P**ersonenführung in U-Bahn**h**öfen: **E**xperimente **u**nd **S**imulationen

- Smoke and fire spread in complex metro stations
  - High dimensional parameter space for fire scenarios, including underground climate
- Pedestrian situation and behavior at evacuation
- Risk analysis regarding human safety



Common Equations for Smoke Spread

- Smoke spread with incompressible Navier–Stokes (NS) equations

$$\nabla \cdot \mathbf{u} = 0$$

$$\rho \left[\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u}\right] + \nabla p - \nabla(2\,\nu\,\epsilon_{ij}(\mathbf{u})) = \mathbf{f}(T)$$

$$\rho \left[\partial_t T + (\mathbf{u} \cdot \nabla) T\right] - 2\,\nu\,\epsilon_{ij}(\mathbf{u}) : \nabla \mathbf{u} - \nabla \cdot \left(\mathrm{Pr}^{-1} \nu \nabla T\right) = q$$

with symmetric gradient $\epsilon_{ij}(\mathbf{u}) = \frac{1}{2}\left[\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right]$

- Turbulence model: Smagorinsky–Lilly LES [6]

$$\nu = \nu_{mol} + \nu_{turb} \qquad \text{with} \qquad \nu_{turb} = (C_s h)^2 \left|\left|\epsilon_{ij}(\mathbf{u})\right|\right|_2$$

Fire Dynamics Simulator

FDS-SMV

Fire **D**ynamics **S**imulator & **Sm**oke**v**iew

- Mainly developed at *National Institute of Standards and Technology* (NIST)
- Provides simulation and visualization framework
- First public release in 2000, written in `Fortran`
- Quasi–standard in fire protection engineering
- Well documented: $\rightarrow$ https://pages.nist.gov/fds-smv/

- Uses Finite Difference Method (FDM) for NS equations
- Elaborated models for combustion and radiation

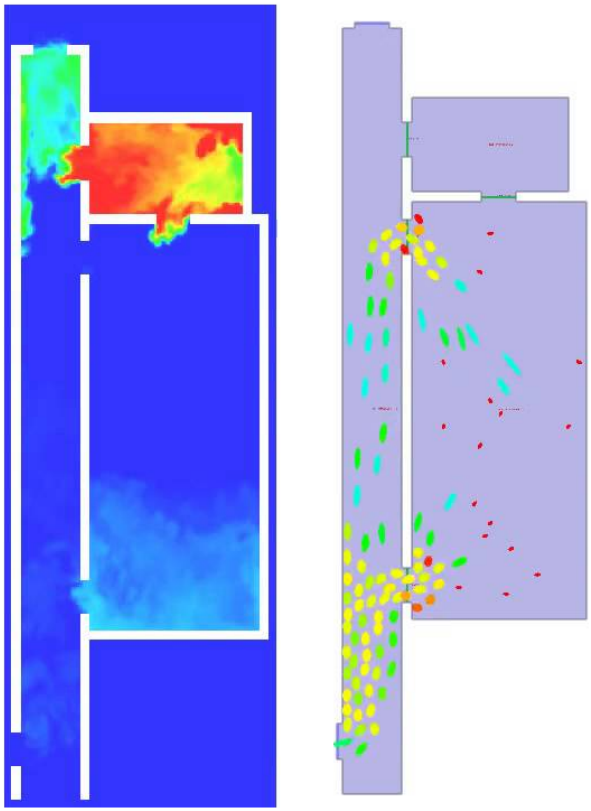
FDS and Evacuation

- Agent–based egress calculation of human beings
- Evacuation module available for `FDS` (`FDS+Evac`)
  - Just for routing, no interaction of agents with fire
- Use locally developed framework `JuPedSim` for more flexibility
- Agent's motion influenced by current smoke situation

  → Smoke sensor

JuPedSim

Juelich **Ped**estrian **Sim**ulator



JuROr

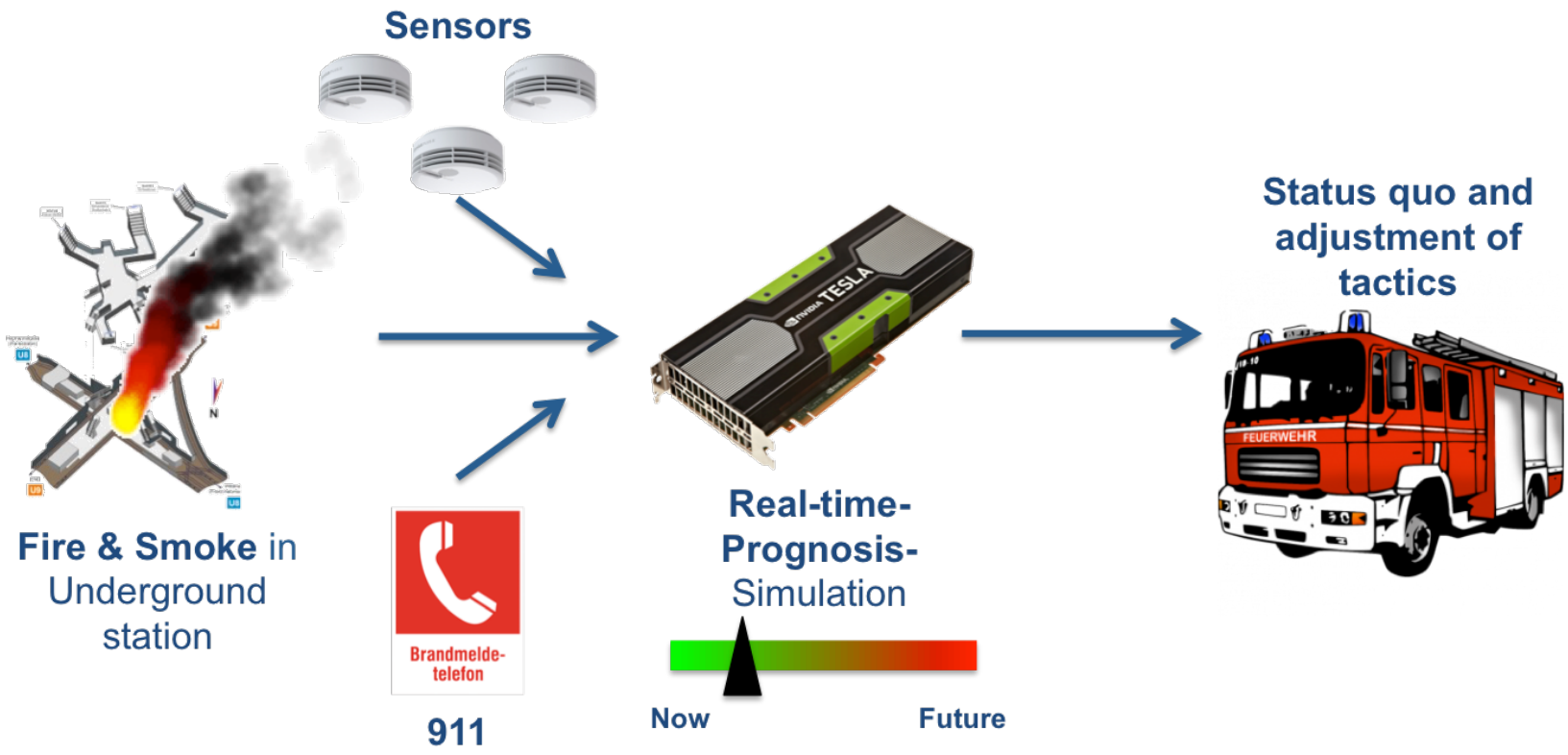
Juelich's **R**eal–time simulation within **Or**pheus

- Aim: Real–time prognosis of fire spread in emergencies
- Help fire fighters to evaluate situation while reaching fire
- Run simulations on (affordable) GPUs

- Uses `OpenACC` API for high flexibility, no external libraries
- Current extent: Flow solver for NS equations with FDM
- Current state: Verification with manufactured solutions

JuROr: Motivation

- Take current situation as boundary values for prognosis



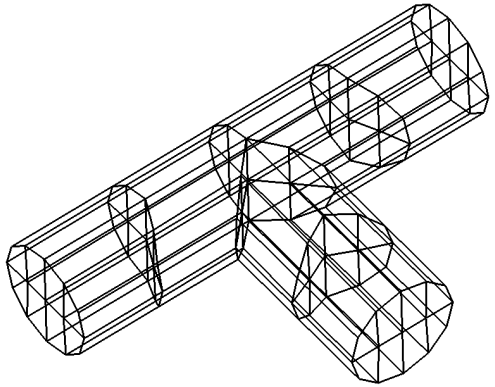
JuFire

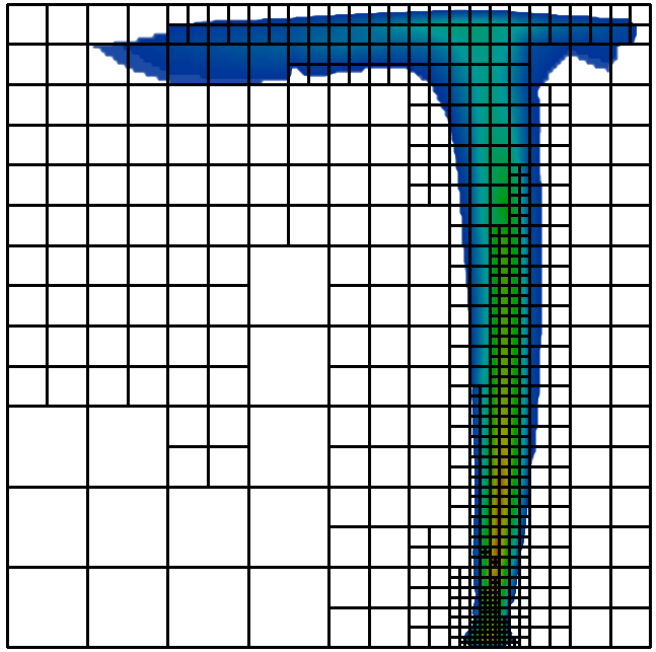
- Investigate applicability of Finite Element Methods (FEM) for fire simulation
- Current state: Verification with manufactured solutions
- Use open–source library deal.II [1]
  - ”Toolbox” for the creation of FEM codes

Differential **E**quations **A**nalysis **L**ibrary

JuFire: Features

- Unstructured grids
- Adaptive refinement in both space and polynomial degree of shape functions (*hp* adaptivity)
- Utilize `CAD` models as manifolds for mesh refinement
  - Use `OpenCASCADE` library
- Continuous and discontinuous Galerkin methods (CG/DG)
- MPI parallelization





Thank you for your attention!

References I


W. Bangerth, D. Davydov, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and D. Wells.
The `deal.II` library, version 8.4.
Journal of Numerical Mathematics, 24, 2016.


Joseph Boussinesq.
Théorie de l'écoulement tourbillonnant et tumultueux des liquides dans les lits rectilignes a grande section (vol.1).
Gauthier–Villars, 1897.

References II


S.L. Glimberg, K. Erleben, and J. Bennetsen.
Smoke simulation for fire engineering using a multigrid method on graphics hardware.
In Hartmut Prautzsch, Alfred Schmitt, Jan Bender, and Matthias Teschner, editors, *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2009)*, pages 11–20. The Eurographics Association, 2009.


N.D. Heavner and L.G Rebholz.
Locally chosen grad–div stabilization parameters for finite elemnt discretizations of incompressible flow problems.
SIAM Undergraduate Research Online (SIURO), 7.

References III


Bernardino Roig.
One–step Taylor Galerkin methods for convection–diffusion problems.
Journal of Computational and Applied Mathematics, 204:95–101, 2007.


Joseph Smagorinsky.
General circulation experiments with the primitive equations.
Monthly Weather Review, 91(3):99–164, March 1963.

References IV


Cedric Taylor and P. Hood.
A numerical solution of the Navier–Stokes equations using the finite element technique.
Computers & Fluids, 1:73–100, 1973.

JÜLICH
FORSCHUNGSZENTRUM

Fire Simulation: Addendum

June 15, 2016 | Marc Fehling | Jülich Supercomputing Centre (JSC)

JuROr: Algorithm

- Algorithm following Glimberg's Ansatz [3]
- Regular grid with data in cell centers
- Finite Difference Method: Central differences in space
- Time discretization: Fractional step method
  - Advection: $\partial_t \mathbf{u}_1 = -(\mathbf{u}_1 \cdot \nabla)\mathbf{u}_1$ Semi–Langrangian method
  - Diffusion: $\partial_t \mathbf{u}_2 = \nu \nabla^2 \mathbf{u}_2$ implicit Jacobi method
  - Sources: $\partial_t \mathbf{u}_3 = \mathbf{f}(T)$ explicit Euler (BD) in time
  - Pressure: $\partial_t \mathbf{u}_4 = -\rho^{-1} \nabla p$ Incompressibility, Multigrid, Jacobi and Projection
- Boussinesq–approximation [2]: $\rho \approx \rho_0 [1 - \beta(T - T_0)]$
  Apply on buoyancy force density: $\mathbf{f}(T) = \rho(T)\mathbf{g}$,
  thus drop second order terms in NS eqs.

JuFire: Algorithm

- Time evolution with backward differentiation formula (BDF)
- Taylor–Hood elements [7]: $(\mathbf{u}, p) \in Q_{k+1}^{\dim} \times Q_k$
- Decoupling of $(\mathbf{u}, p)$ by projection scheme
  - Leads to the pressure–Poisson equation
- Stabilization of momentum equation
  - Taylor–Galerkin stabilization [5] for Taylor–Hood elements
    - Additional diffusion in flow direction
  - Grad–div stabilization [4] to enforce $\nabla \cdot \mathbf{u} = 0$
- Neumann series for fast matrix assembly
- Boussinesq approximation for buoyancy force density